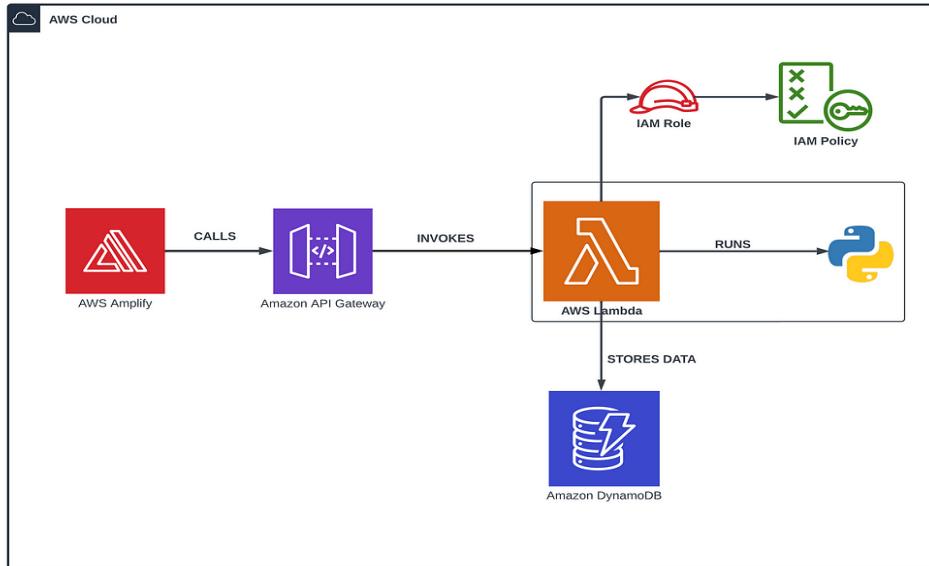


# Serverless Web App Development



## Project Overview:

This project is a web application that calculates the area of a rectangle. Users can input the length and width of a rectangle, and the application will display the calculated area. The application is built using AWS services such as AWS Amplify, API Gateway, Lambda, and DynamoDB.

## Prerequisites:

AWS account

Frontend web application (Index.html file)

## Setup Instructions:

- Setting up Dynamo DB
- Setting up Lambda
- Setting up API gateway
- Setting up Frontend web application
- Setting up AWS amplify

Sign into your AWS console

AWS (Amazon Web Services) is a comprehensive cloud computing platform offering a wide range of services for computing, storage, and networking.

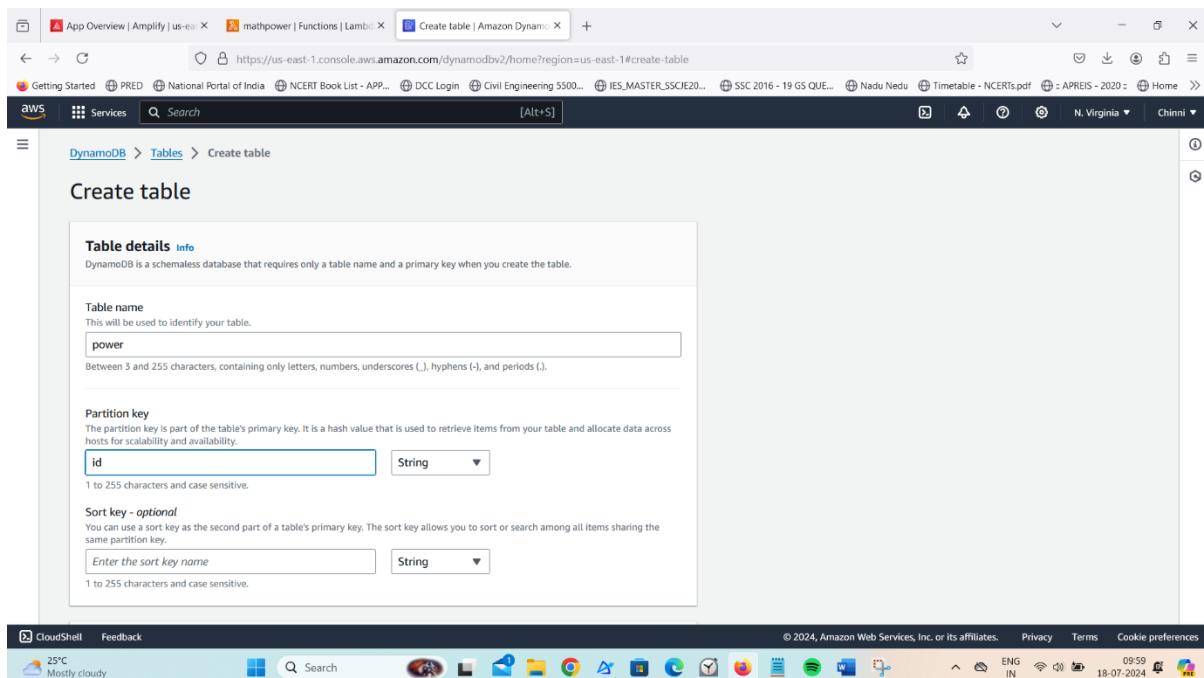
Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

## Step1: Setting up Dynamo DB

On the services search box, search for ‘DynamoDB’ and select the DynamoDB service.

Click on create table.

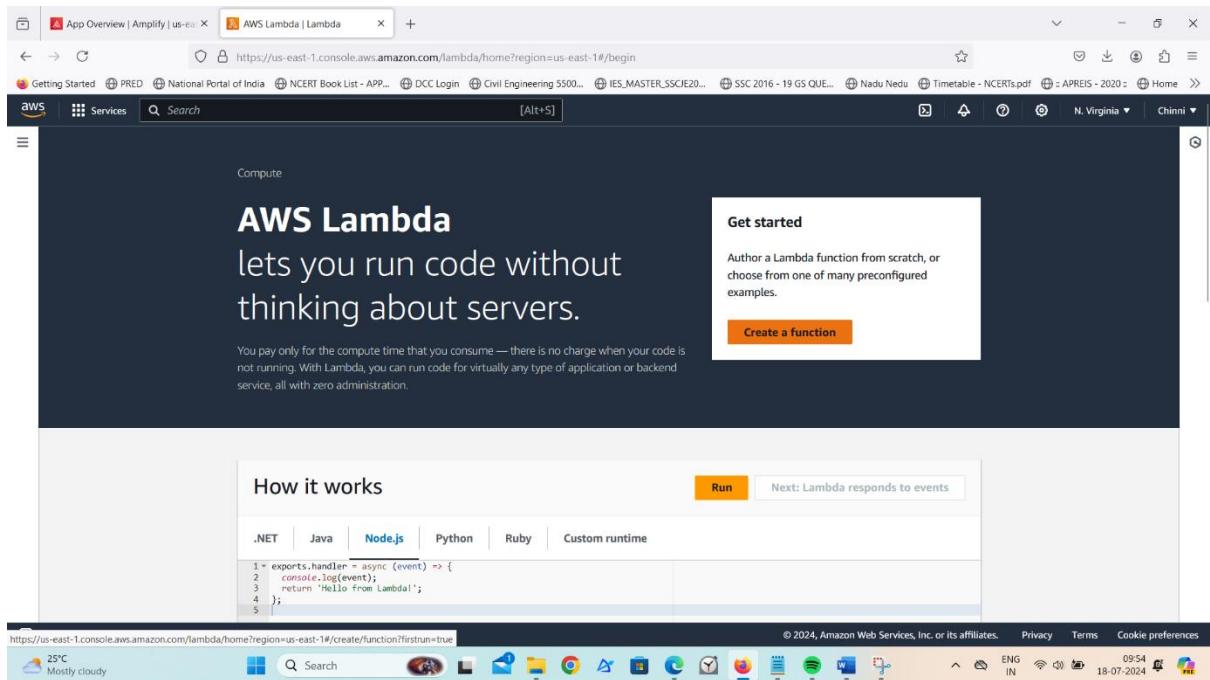
Give the table a name, for ‘Partition key’ input ‘ID’. Leave the rest as default, scroll to the bottom and click on ‘Create table’



AWS Lambda is a serverless compute service that automatically runs your code in response to events and scales the underlying compute resources as needed, without the need to provision or manage servers.

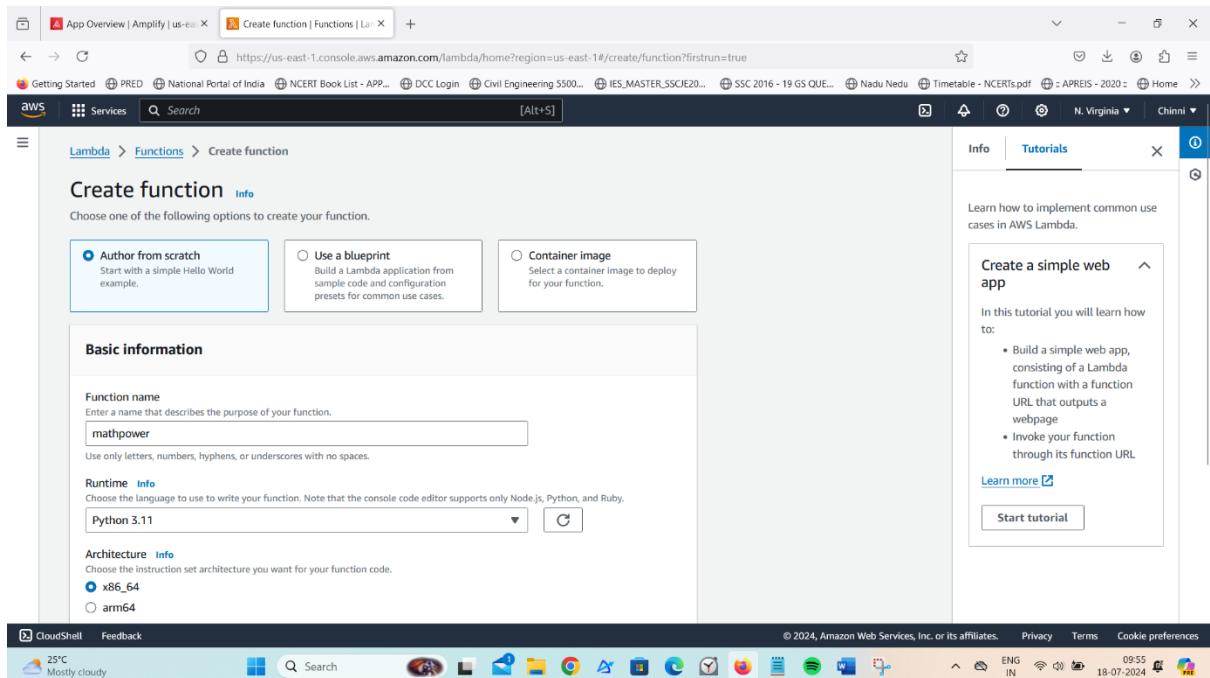
## Step2: Setting up Lambda

On the AWS console search bar, type ‘Lambda’ and select the Lambda service.



Click on ‘Create function’.

Give the Function name, The Runtime (Latest Python), then scroll down and click on ‘Create function’



Copy the following Lambda function onto your lambda\_function.py file. Please give the DynamoDB name (as created above).

```
import json
import boto3
from time import gmtime, strftime

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('power')

now = strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())

def lambda_handler(event, context):
    try:
        length = int(event['length'])
        width = int(event['width'])
        Area = length * width

        response = table.put_item(
            Item={
                'ID': str(Area),
                'LatestGreetingTime': now,
            }
        )
        return {
            'statusCode': 200,
            'body': json.dumps('Your result is ' + str(Area))
    
```

```
}
```

```
except ValueError:
```

```
    return {
```

```
        'statusCode': 400,
```

```
        'body': json.dumps('Error: length and width must be integers.')
```

```
}
```

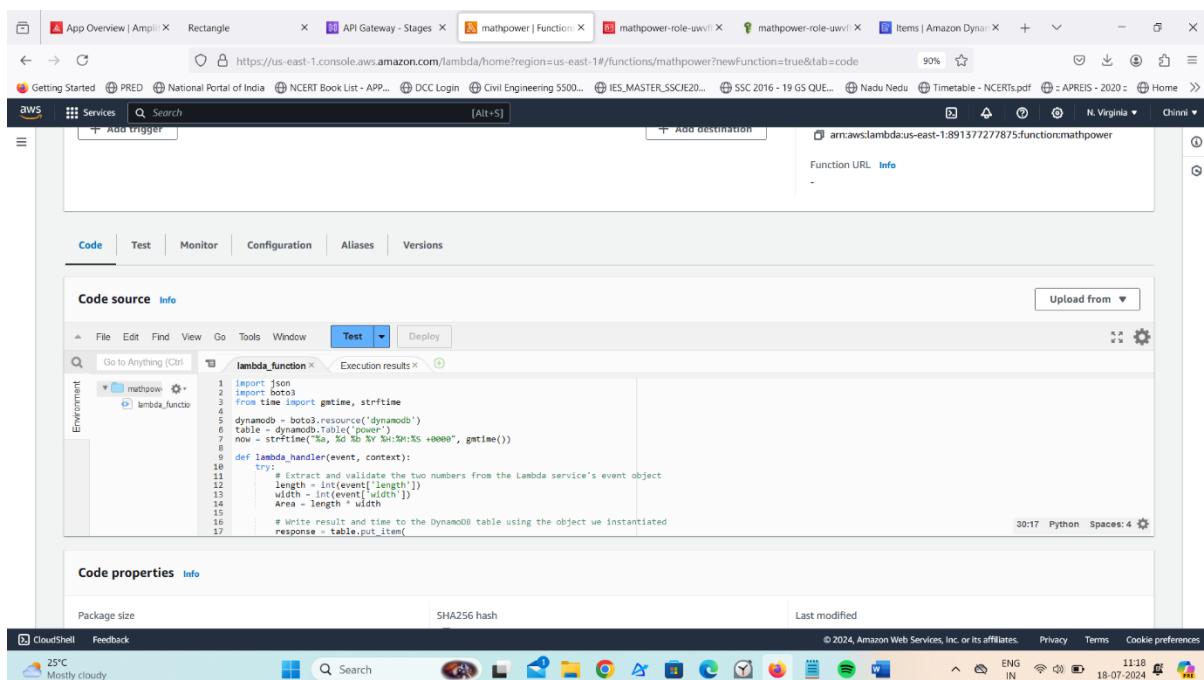
```
except Exception as e:
```

```
    return {
```

```
        'statusCode': 500,
```

```
        'body': json.dumps('Error: ' + str(e))
```

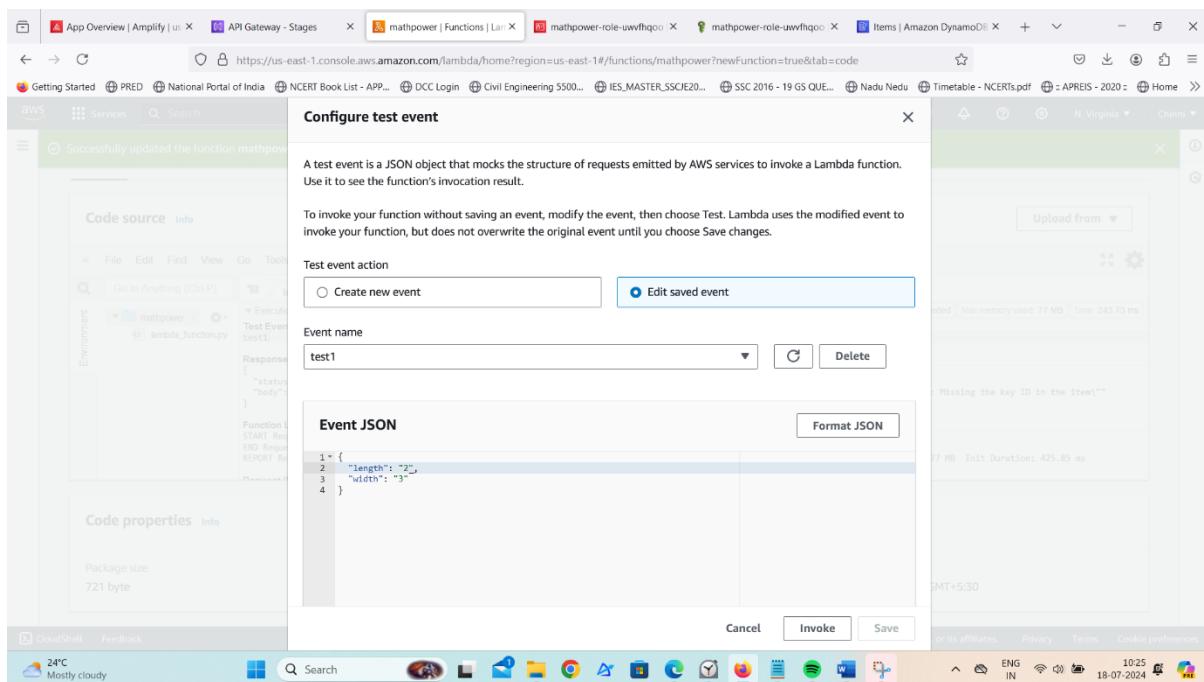
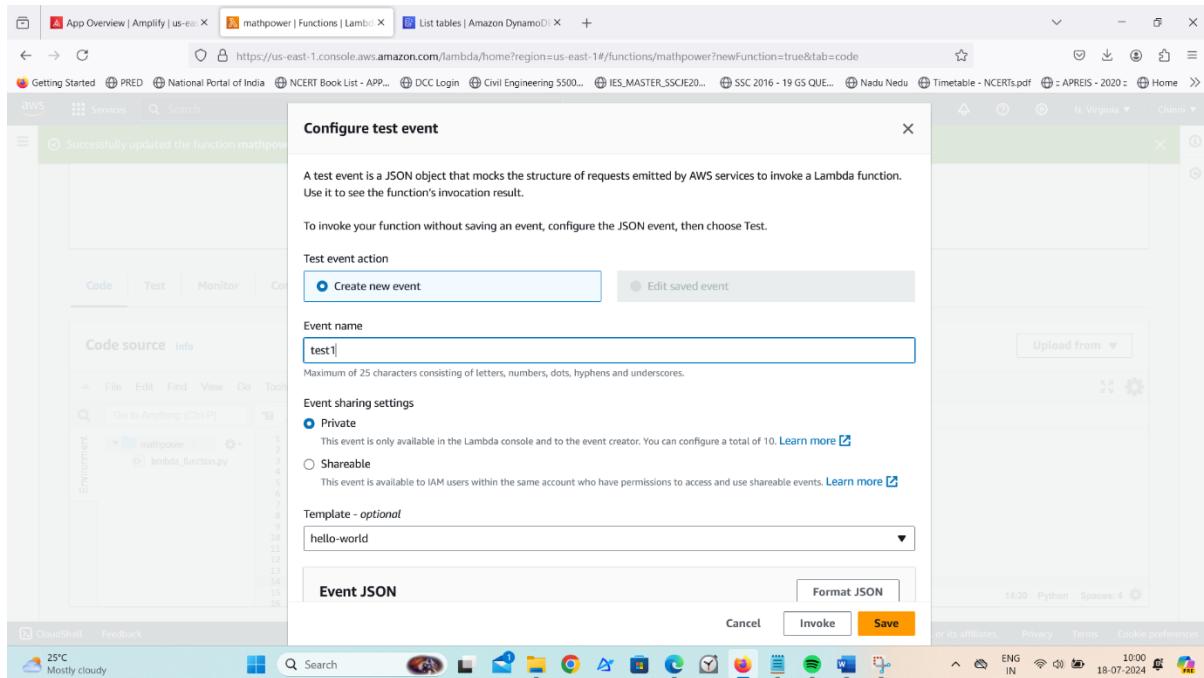
```
}
```



Click on deploy. Then click on test. We will get a window open then give event name and in event give the values of length and width.

Click on save and click on test.

## We get our result ( $2*3=6$ )



At first test we get an error like access denied for the click on lambda configurations tab in that click on permissions tab.

An IAM role in AWS (Identity and Access Management) defines a set of permissions that grant access to AWS resources. It is used to delegate access to users, applications, or services, ensuring secure and controlled interactions with AWS services and resources.

In permissions tab we can see the IAM role click on that.

The screenshot shows the AWS Lambda console. In the left sidebar, 'Permissions' is selected under the 'Execution role' section. The main area displays the 'Role name' as 'mathpower-role-uvwxyzqoo'. Below it, the 'Resource summary' section shows that the function has permission to access Amazon CloudWatch Logs, with 3 actions and 2 resources. The 'By resource' tab is selected. The bottom of the screen shows the Windows taskbar with various pinned icons.

The screenshot shows the AWS IAM console. In the left sidebar, 'Roles' is selected under 'Access management'. The main area shows the 'Permissions' tab for the 'mathpower-role-uvwxyzqoo' role. It lists one managed policy, 'AWSLambdaBasicExecutionRole-289f03b...', which is customer-managed. There is also a section for generating a policy based on CloudTrail events. The bottom of the screen shows the Windows taskbar with various pinned icons.

It will open like this, now click on add permissions in that we see create inline policy. Click on it.

Click on JSON in that search for dynamo DB in all services tab.

Give policies like put item, delete item, update item, get item, scan, query.

```

1▼ {
2    "Version": "2012-10-17",
3    "Statement": [
4        {
5            "Sid": "Statement1",
6            "Effect": "Allow",
7            "Action": [],
8            "Resource": []
9        }
10    ]
11 }

```

The screenshot shows the 'Specify permissions' step of creating a new IAM policy. The 'Policy editor' section contains the JSON code above. To the right, there's a sidebar for 'Edit statement' and 'Add actions'. Under 'Add actions', 'All services > DynamoDB' is selected, and a list of actions is shown, with 'All actions (dynamodb:\*)' checked. Other options like 'ListBackups' and 'ListImports' are also listed.

At resources copy our dynamo DB table arn (amazon resource name).

Click on our table we get general information under that we have additional info drag that tab we see our table arn paste that on policy.

Partition key	Sort key	Capacity mode	Table status
id (String)	-	Provisioned	<span>Active</span>

**General information**

- Alarms: No active alarms
- Point-in-time recovery (PITR): Info
- Resource-based policy: Info
- Table status: Active

**Additional info**

Table class	Indexes	DynamoDB stream	Time to Live (TTL)
DynamoDB Standard	0 globals, 0 locals	Off	Off

Amazon Resource Name (ARN)  
arn:aws:dynamodb:us-east-1:891377277875:table/power

The screenshot shows the 'DynamoDB' service page for a table named 'power'. On the left, there's a navigation menu with 'Tables' selected. The main area displays 'General information' and 'Additional info' sections. The 'Amazon Resource Name (ARN)' field is explicitly highlighted with a red box.

The screenshot shows the AWS IAM 'Create policy' interface. The URL is <https://us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/mathpower-role-uvwxyzqo/createPolicy>. The page title is 'Create policy | IAM | Global'. The left sidebar shows 'Step 1: Specify permissions' and 'Step 2: Review and create'. The main area is titled 'Specify permissions' with the sub-section 'Policy editor'. The JSON code in the editor is:

```
1 Version: "2012-10-17",
2 Statement: [
3   {
4     Sid: "Statement1",
5     Effect: "Allow",
6     Action: [
7       "dynamodb:PutItem",
8       "dynamodb:DeleteItem",
9       "dynamodb:GetItem",
10      "dynamodb:Scan",
11      "dynamodb:Query",
12      "dynamodb:UpdateItem"
13    ],
14    Resource: "arn:aws:dynamodb:us-east-1:891377277875:table/power"
15  }
16 ]
17
18 ]
```

To the right of the editor, there's a 'Visual' tab, a 'JSON' tab (which is selected), and an 'Actions' dropdown. Below the editor, there's a section titled 'Edit statement' with a link 'Select a statement' and a button '+ Add new statement'. The bottom of the screen shows a Windows taskbar with various icons and system status.

Click on next.

Give policy name and click on create policy.

The screenshot shows the AWS IAM 'Review and create' interface. The URL is <https://us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/mathpower-role-uvwxyzqo/createPolicy>. The page title is 'Create policy | IAM | Global'. The left sidebar shows 'Step 1: Specify permissions' and 'Step 2: Review and create'. The main area is titled 'Review and create' with the sub-section 'Policy details'. The 'Policy name' field contains 'dynamodbper'. The 'Permissions defined in this policy' section shows a table with one row for 'DynamoDB' with access level 'Limited: Read, Write' and resource 'region| string like [us-east-1, TableName| string like [power]'. There are 'Edit' and 'Show remaining 419 services' buttons. At the bottom, there are 'Cancel', 'Previous', and 'Create policy' buttons. The bottom of the screen shows a Windows taskbar with various icons and system status.

The screenshot shows the AWS Identity and Access Management (IAM) console. The left sidebar includes sections for Dashboard, Access management (User groups, Roles, Policies, Identity providers, Account settings), and Access reports (Access Analyzer, External access, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies). The main content area is titled 'Permissions policies (2)' and lists two entries:

Policy name	Type	Attached entities
AWSLambdaBasicExecutionRole-289f03b...	Customer managed	1
dynamodbper	Customer inline	0

Below this, there's a section for 'Permissions boundary (not set)' and a 'Generate policy based on CloudTrail events' section with a 'Generate policy' button.

Now go to lambda click on test we get result.

The screenshot shows the AWS Lambda console for the function 'mathpower'. The 'Code source' tab is active, showing the code structure with a file named 'lambda\_function.py' containing:

```
def lambda_handler(event, context):
    response = {
        "statusCode": 200,
        "body": "\'Your result is 6\'"
    }
    return response
```

The 'Test' tab is selected, showing the execution results for a test event named 'test1'. The results table shows:

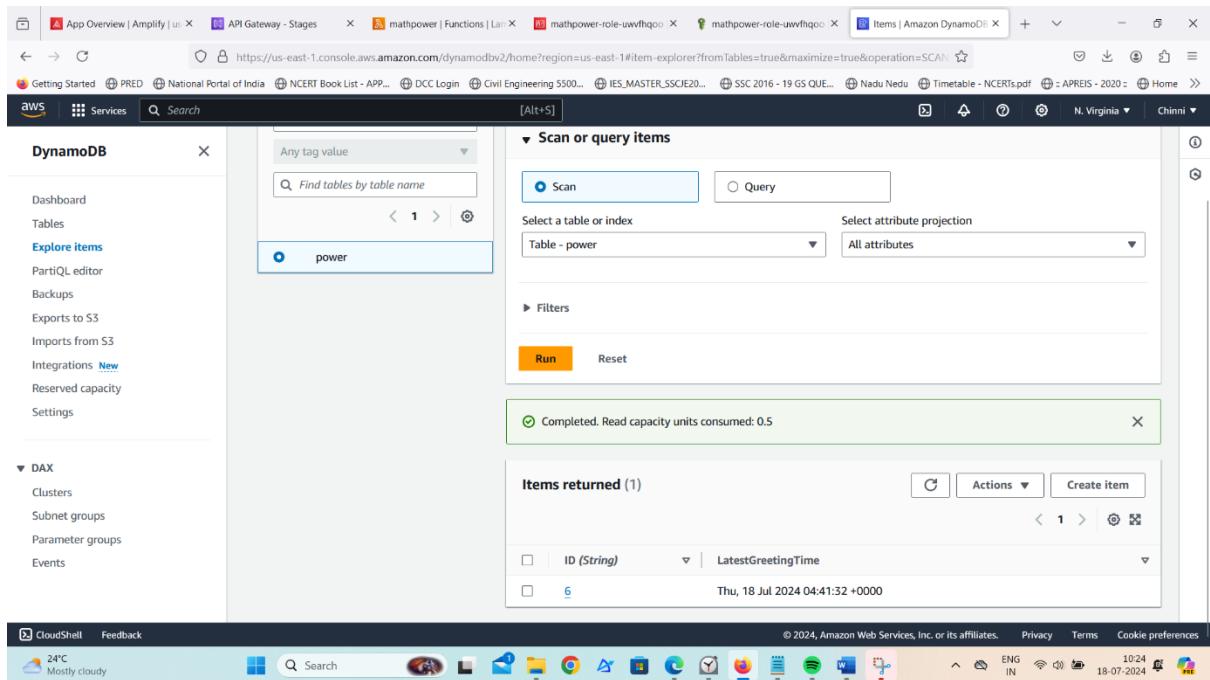
Execution results
Test Event Name test1
Response {"statusCode": 200, "body": "\'Your result is 6\'"} Function Logs START RequestId: 19a5d5ee-ac55-4a8c-846f-2734f7e20deb Version: \$LATEST END RequestId: 19a5d5ee-ac55-4a8c-846f-2734f7e20deb REPORT RequestId: 19a5d5ee-ac55-4a8c-846f-2734f7e20deb Duration: 164.10 ms Billed Duration: 165 ms Memory Size: 128 MB Max Memory Used: 79 MB

The 'Code properties' tab shows:

Package size	SHA256 hash	Last modified
714 byte	tzbzu+d7OVW7LMJYBrjM67l/Wpb44CqOA5S4pSl85s=	July 18, 2024 at 09:55 AM GMT+5:30

The 'Runtime settings' tab is also present.

we can see our data is saved on our created dynamo DB table.



### Step 3: Setting up API gateway

API Gateway in AWS is a fully managed service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale. It acts as a front door for applications to access data, business logic, or functionality from backend services, including Lambda functions, AWS services, or other HTTP endpoints.

- On the AWS services search box, enter ‘API’ and select ‘API Gateway’ that appears
- In the list for ‘Choose an API type’, select ‘Build’ for ‘REST API’

The screenshot shows the AWS CloudShell interface with the REST API creation wizard. The top navigation bar includes tabs for Amplify, API Gateway - Create API, mathpower Functions | Lambda, mathpower-role-uvvfhqoo, mathpower-role-uvvfhqoo, Items | Amazon DynamoDB, and Home. The main content area has two sections: 'REST API' and 'REST API Private'. Both sections contain descriptions, compatibility notes (Lambda, HTTP, AWS Services), and 'Import' and 'Build' buttons. The status bar at the bottom shows CloudShell, Feedback, a search bar, and various system icons.

A REST API is an architectural style for networked applications using HTTP requests to access and manipulate resources via URLs.

Choose the ‘REST’ protocol for the API, select ‘New API’ under ‘Create new API’ and give the API a name, then click on ‘Create API’

The screenshot shows the 'Create REST API' form in the AWS CloudShell interface. The 'API details' section includes options for 'New API' (selected), 'Clone existing API', 'Import API', and 'Example API'. The 'API name' field is set to 'api-power'. The 'Description - optional' field is empty. The 'API endpoint type' section indicates 'Edge-optimized' deployment. At the bottom are 'Cancel' and 'Create API' buttons. The status bar at the bottom shows CloudShell, Feedback, a search bar, and various system icons.

An API POST method is used to send data to a server to create or update a resource.

**Create method**

**Method details**

Method type: POST

Integration type:

- Lambda function**: Integrate your API with a Lambda function.
- HTTP**: Integrate with an existing HTTP endpoint.
- Mock**: Generate a response based on API Gateway mappings and transformations.
- AWS service**: Integrate with an AWS Service.
- VPC link**: Integrate with a resource that isn't accessible over the public internet.

CloudShell Feedback 24°C Mostly cloudy Search ENG IN 10:20 18-07-2024 N. Virginia Chinni

Send the request to your Lambda function as a structured event.

**Lambda function**: Provide the Lambda function name or alias. You can also provide an ARN from another account. us-east-1 arm:aws:lambda:us-east-1:891377277875:function:mat

**Grant API Gateway permission to invoke your Lambda function.**: To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

**Default timeout**: The default timeout is 29 seconds.

**Method request settings**

**URL query string parameters**

**HTTP request headers**

**Request body**

CloudShell Feedback 24°C Mostly cloudy Search ENG IN 10:20 18-07-2024 N. Virginia Chinni

**API Gateway**

**APIs**: Custom domain names, VPC links

**API: api-power**: Resources, Stages, Authorizers, Gateway responses, Models, Resource policy, Documentation, Dashboard, API settings

**Usage plans**, **API keys**, **Client certificates**

**Resources**: Create resource / POST / - POST - Method execution

ARN: arm:aws:execute-api:us-east-1:891377277875:q7xum4aqb/\*/POST/

Resource ID: 9yyzb0l2ck

Method request → Integration request → Integration response ← Method response ← Integration response ← Method response

**Method request settings**

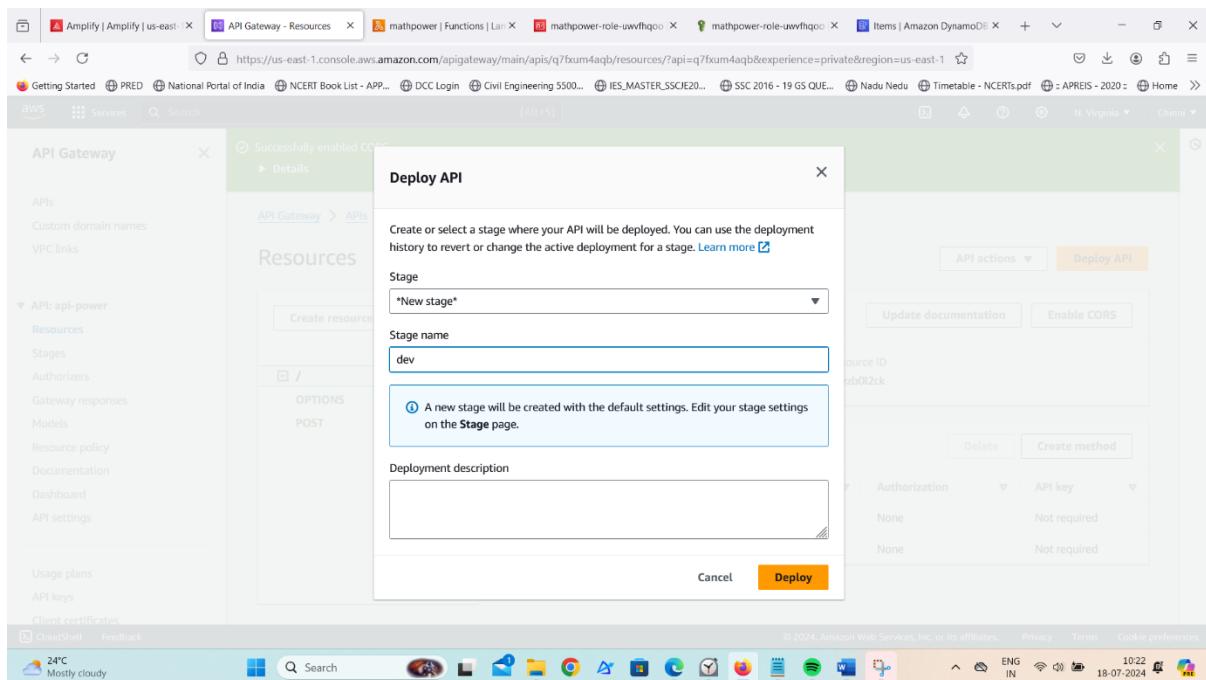
API actions Deploy API

CloudShell Feedback 24°C Mostly cloudy Search ENG IN 10:21 18-07-2024 N. Virginia Chinni

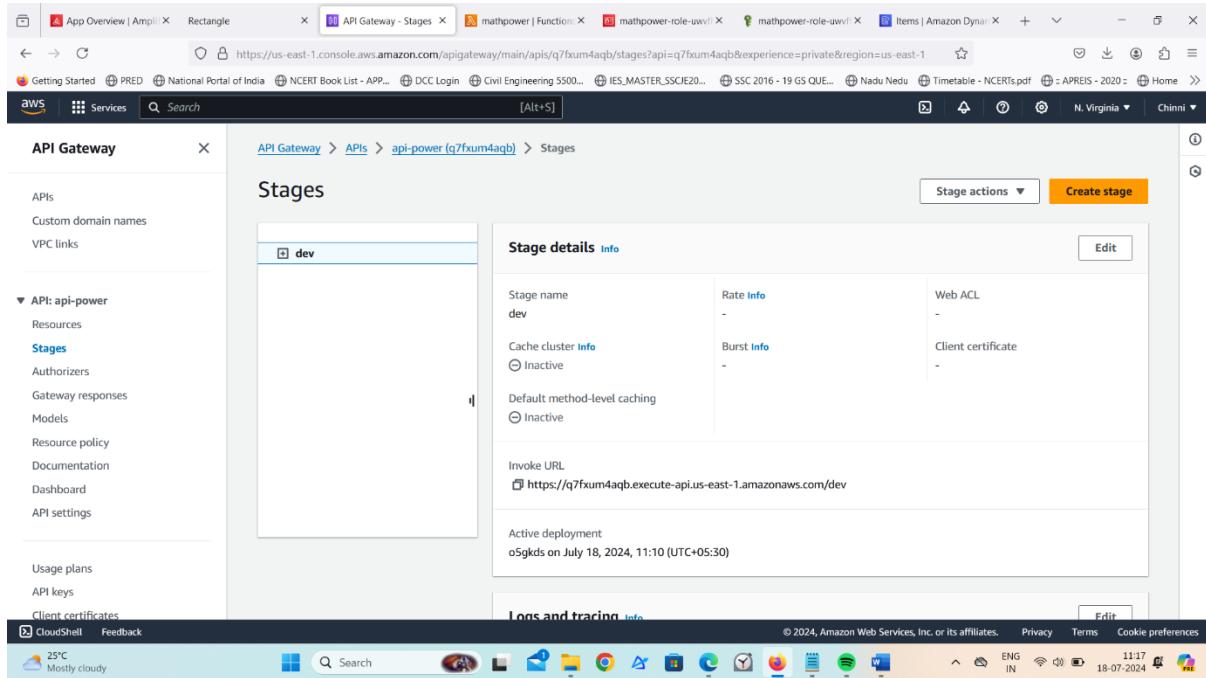
The screenshot shows the AWS API Gateway Resources page. On the left, a sidebar for API: api-power lists Resources, Stages, Authorizers, Gateway responses, Models, Resource policy, Documentation, Dashboard, and API settings. Under Resources, it shows a single resource at path '/'. The main panel displays 'Resource details' with Path: / and Resource ID: 9yyzb0l2ck. Below is a table for 'Methods (1)'. It has columns for Method type (POST), Integration type (Lambda), Authorization (None), and API key (Not required). A 'Create method' button is also present.

CORS (Cross-Origin Resource Sharing) in API Gateway is a mechanism that allows resources on a web page to be requested from another domain outside the domain from which the first resource was served, enhancing the security of web applications by controlling access to resources.

The screenshot shows the AWS API Gateway Enable CORS page for the '/' resource. It includes sections for CORS settings, Gateway responses (Default 4XX checked, Default 5XX unchecked), Access-Control-Allow-Methods (OPTIONS and POST checked), Access-Control-Allow-Headers (Content-Type, X-Amz-Date, Authorization, X-Api-Key, X-Amz-Security-Token), and Access-Control-Allow-Origin (a dropdown menu with a placeholder '\*'). A 'Additional settings' link is at the bottom.



A stage in API Gateway represents a deployment environment (e.g., development, production) where API configurations and resources can be managed, tested, and accessed.



## **Step4:** Setting up frontend web application

Create a new file in desktop and keep the below code in it.

```
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Rectangle</title>

<!-- Styling for the client UI -->

<style>

    h1 { color: #FFFFFF; font-family: system-ui; margin-left: 20px; }

    body { background-color: #222629; }

    label { color: #86C232; font-family: system-ui; font-size: 20px; margin-left: 20px; margin-top: 20px; }

    button { background-color: #86C232; border-color: #86C232; color: #FFFFFF; font-family: system-ui; font-size: 20px; font-weight: bold; margin-left: 30px; margin-top: 20px; width: 140px; }

    input { color: #222629; font-family: system-ui; font-size: 20px; margin-left: 10px; margin-top: 20px; width: 100px; }

</style>

<script>

    var callAPI = (length, width) => {

        var myHeaders = new Headers();

        myHeaders.append("Content-Type", "application/json");

        // Ensure the length and width are integers
    }
}
```

```
length = parseInt(length);

width = parseInt(width);

if (isNaN(length) || isNaN(width)) {

    alert('Length and width must be valid numbers');

    return;

}

var raw = JSON.stringify({ "length": length, "width": width });

var requestOptions = { method: 'POST', headers: myHeaders, body: raw,
redirect: 'follow' };

fetch("https://q7fxum4aqb.execute-api.us-east-1.amazonaws.com/dev",
requestOptions)

.then(response => response.text())

.then(result => alert(JSON.parse(result).body))

.catch(error => console.log('error', error));

}

</script>

</head>

<body>

<h1>AREA OF A RECTANGLE!</h1>

<form>

<label>Length:</label>
```

```

<input type="text" id="length">

<label>Width:</label>

<input type="text" id="width">

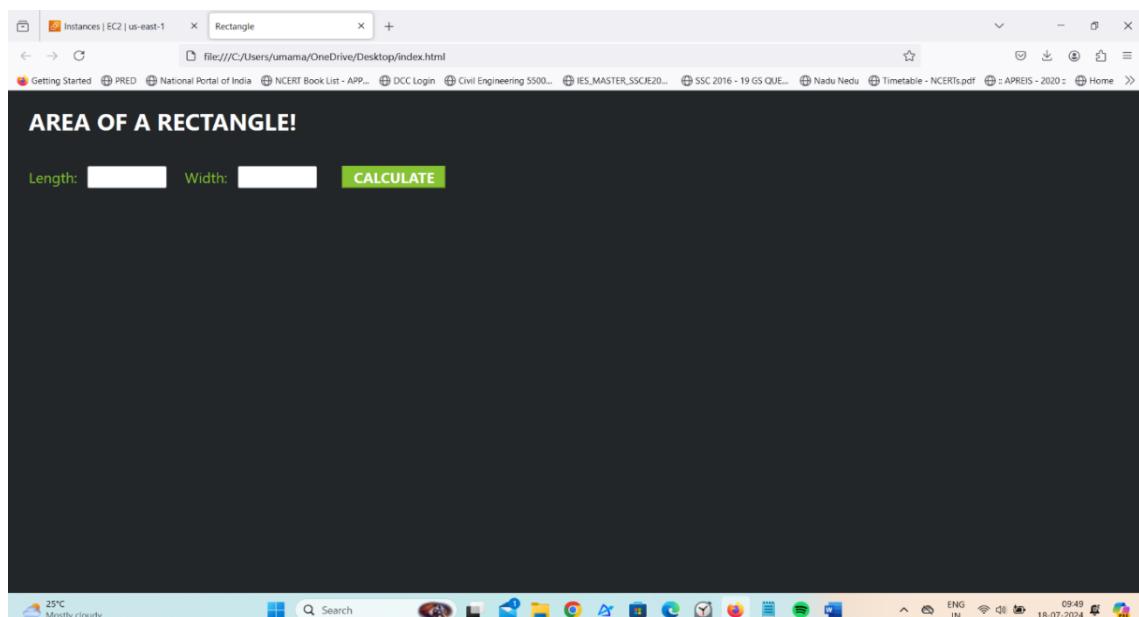
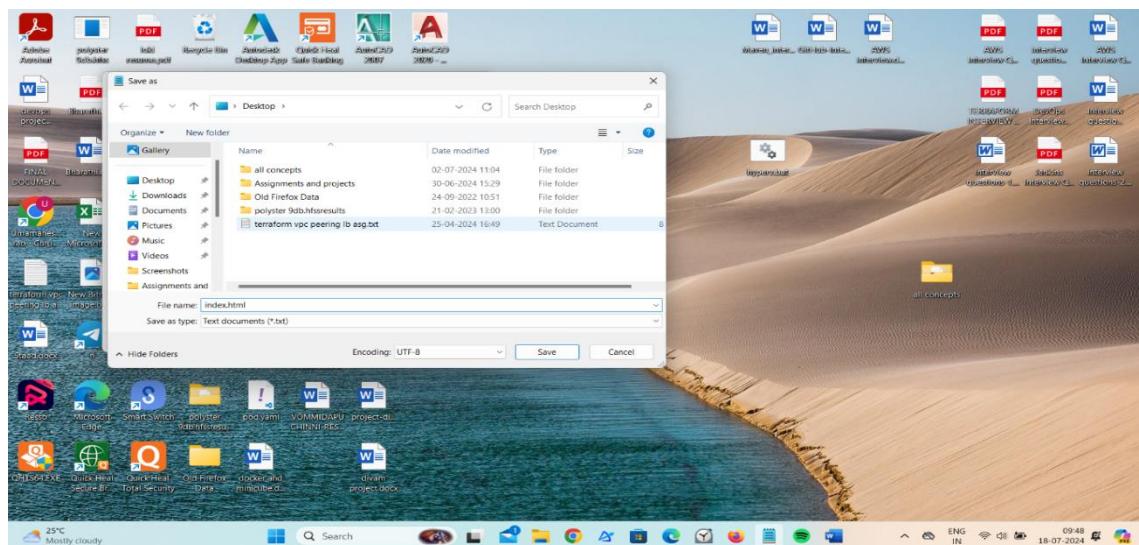
<button type="button"
onclick="callAPI(document.getElementById('length').value,
document.getElementById('width').value)">CALCULATE</button>

</form>

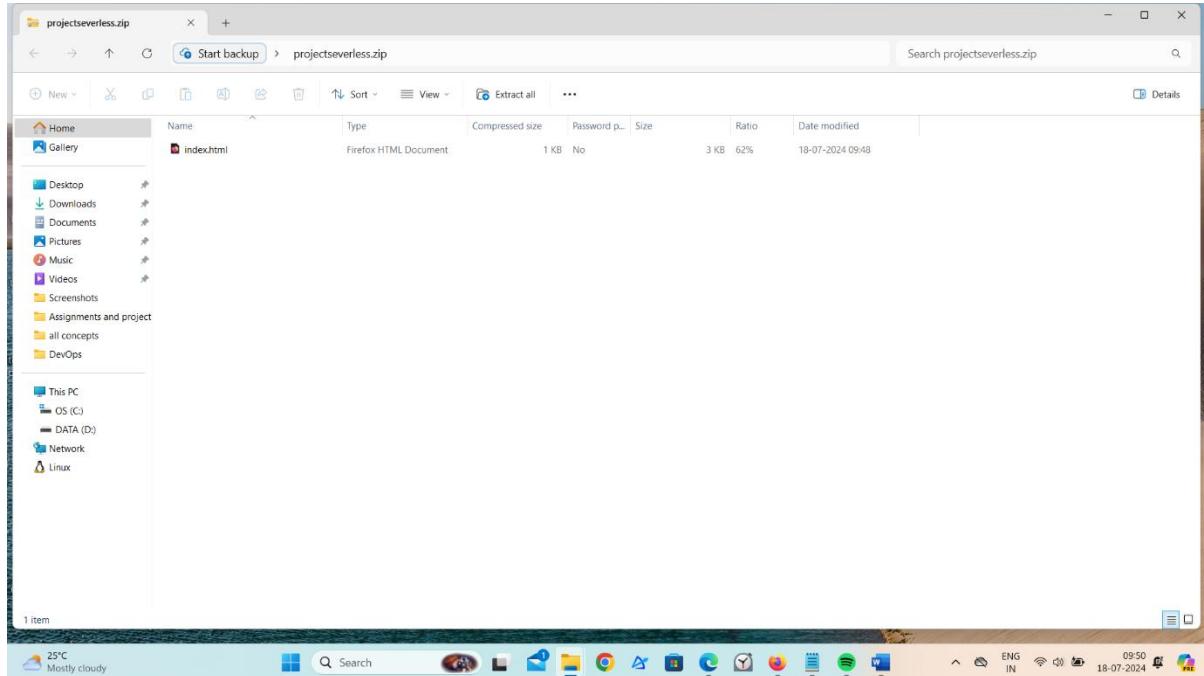
</body>

</html>

```

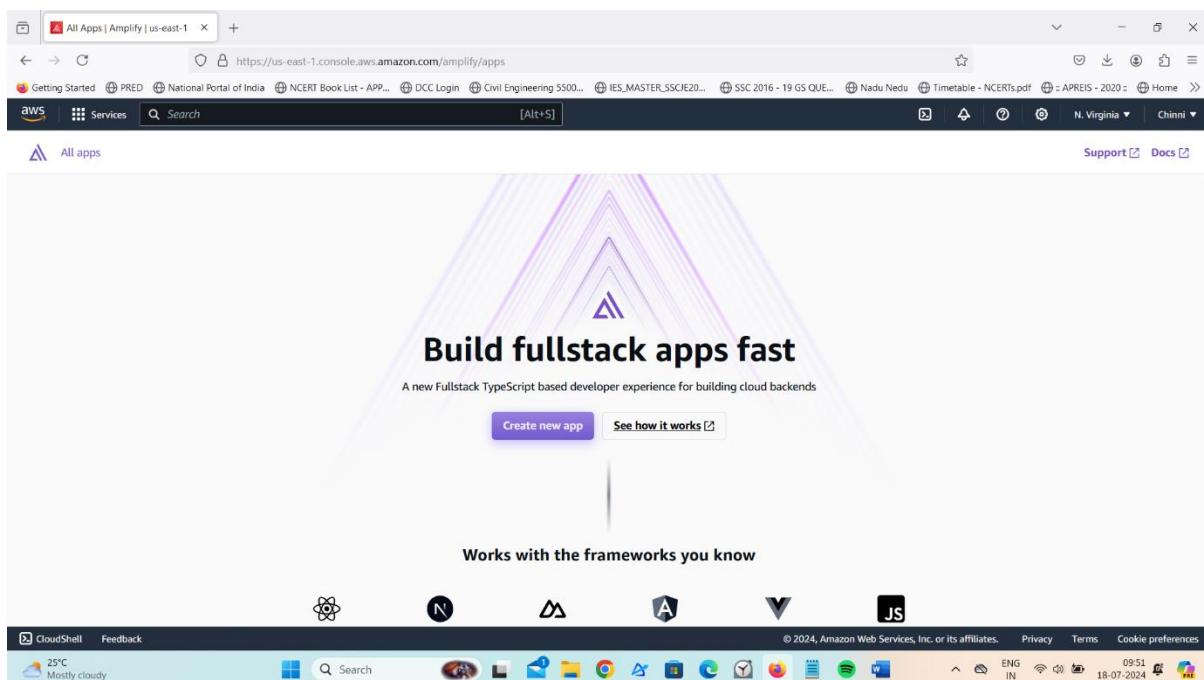


Create a new zip folder on desktop and move that index.html file into that zip folder.

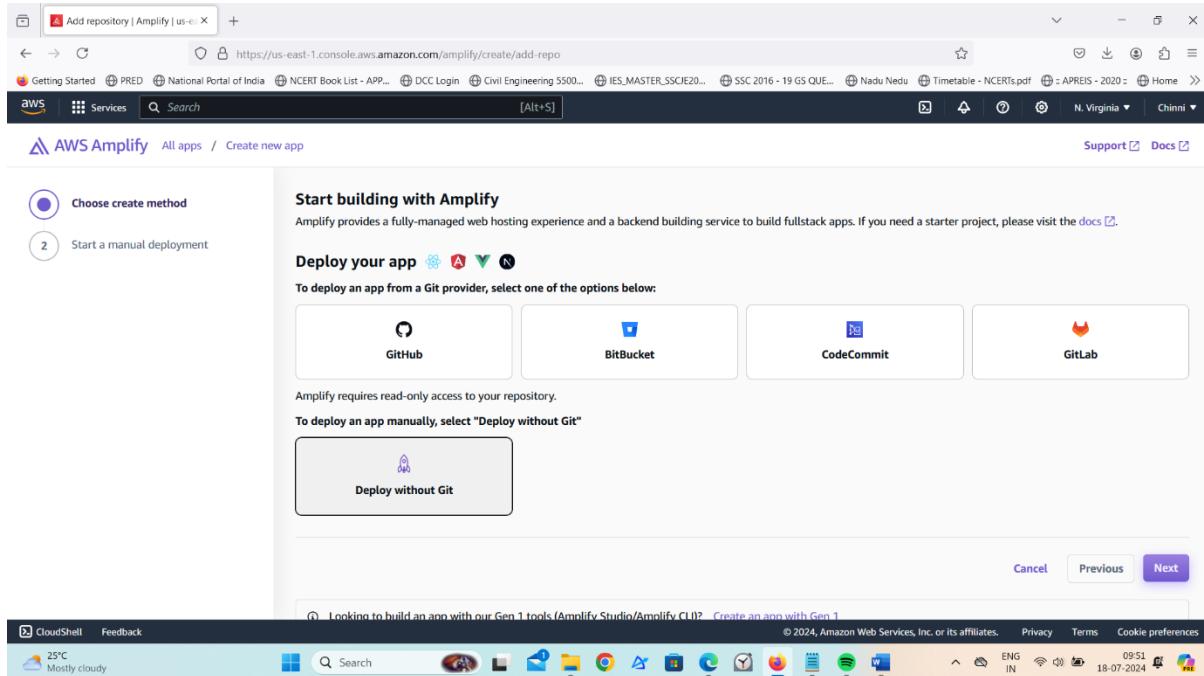


AWS Amplify is a set of tools and services provided by Amazon Web Services (AWS) that enables developers to build scalable full-stack applications quickly. It includes features for frontend web and mobile development, backend services like authentication, APIs, databases, storage, and hosting, all managed through a single unified platform.

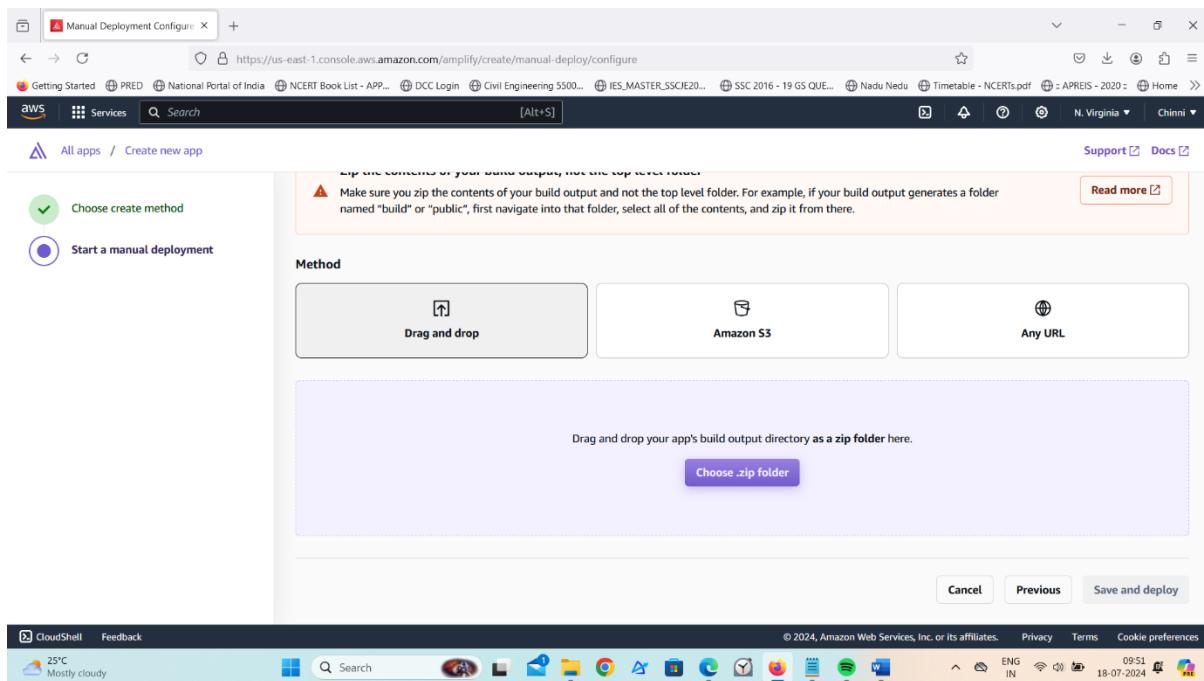
Search AWS Amplify in AWS search tab. Open it, click on create new app.



Select deploy without git, because we kept our code in our own zip folder. Click on next.



Click on choose.zip folder. Choose our zip folder. Click on open.



Click on save and deploy.

Wait until its available.

The screenshot shows the AWS Amplify console interface for creating a new app named 'staging'. The 'Choose create method' option is selected. A warning message states: 'Zip the contents of your build output, not the top level folder. Make sure you zip the contents of your build output and not the top level folder. For example, if your build output generates a folder named "build" or "public", first navigate into that folder, select all of the contents, and zip it from there.' Below this, there are three methods: 'Drag and drop' (selected), 'Amazon S3', and 'Any URL'. A file named 'projectseverless.zip' is uploaded. At the bottom right, there are 'Cancel', 'Previous', and 'Save and deploy' buttons.

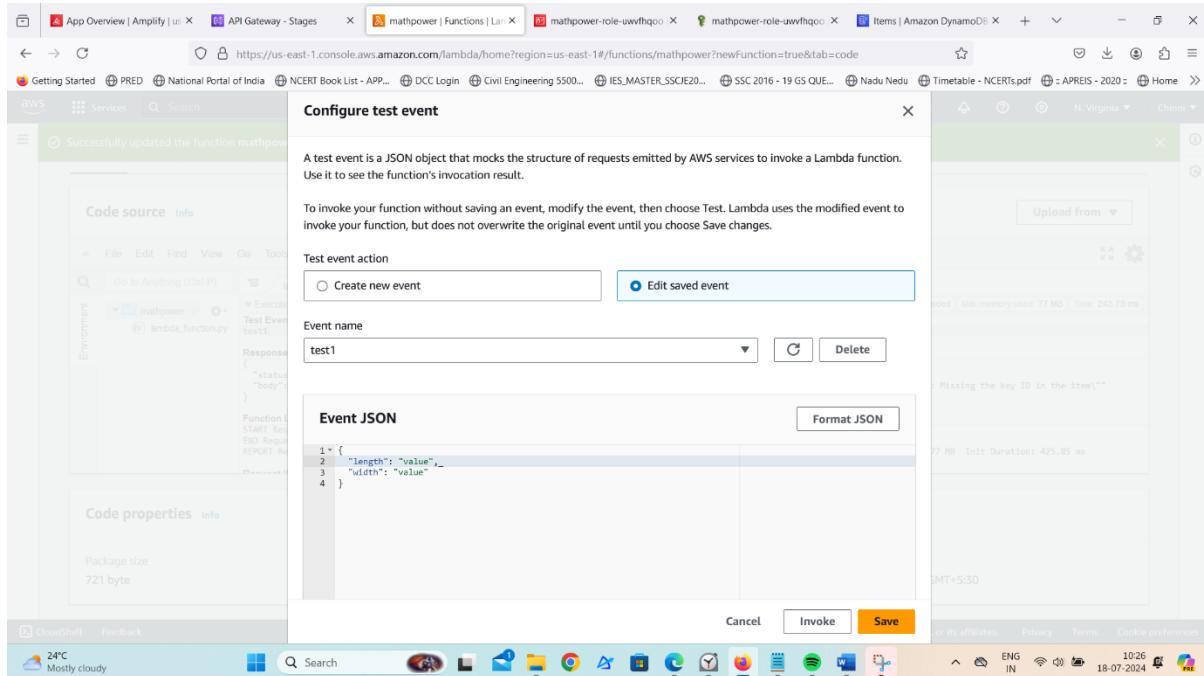
  

The screenshot shows the AWS Amplify console interface for the app 'app9854'. The 'Overview' tab is selected. It shows the app ID 'd173v0xm1x7fnr'. The 'Production branch' section shows the 'staging' branch is deployed. The domain URL is listed as 'https://staging.d173v0xm1x7fnr.amplifyapp.com'. There is a 'Visit deployed URL' button. Other branches are listed below, with a note that none have been added.

Click on domain we see our frontend web app.

The screenshot shows a web browser window with the title 'App Overview | Amplify | us-east-1'. The page content is a simple form titled 'AREA OF A RECTANGLE!' with fields for 'Length' and 'Width' and a 'CALCULATE' button. The background is dark, and the text is white.

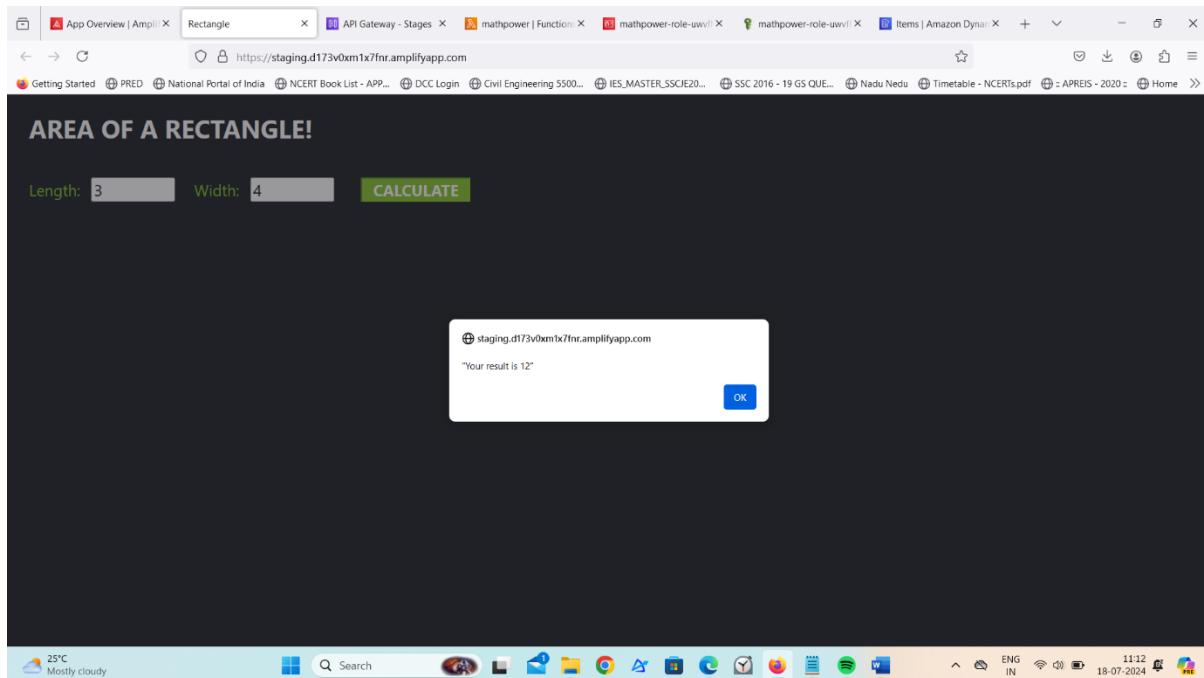
Now go to lambda click on test side button we see configure event test click on it and change to values in EVENT JSON. Click on save.

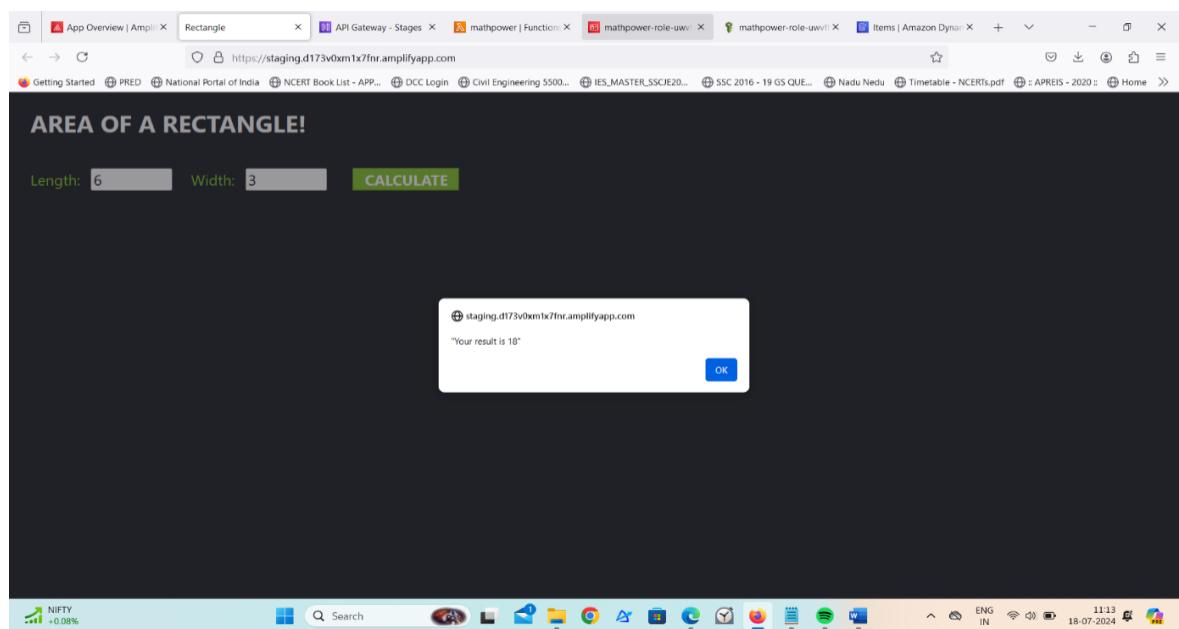
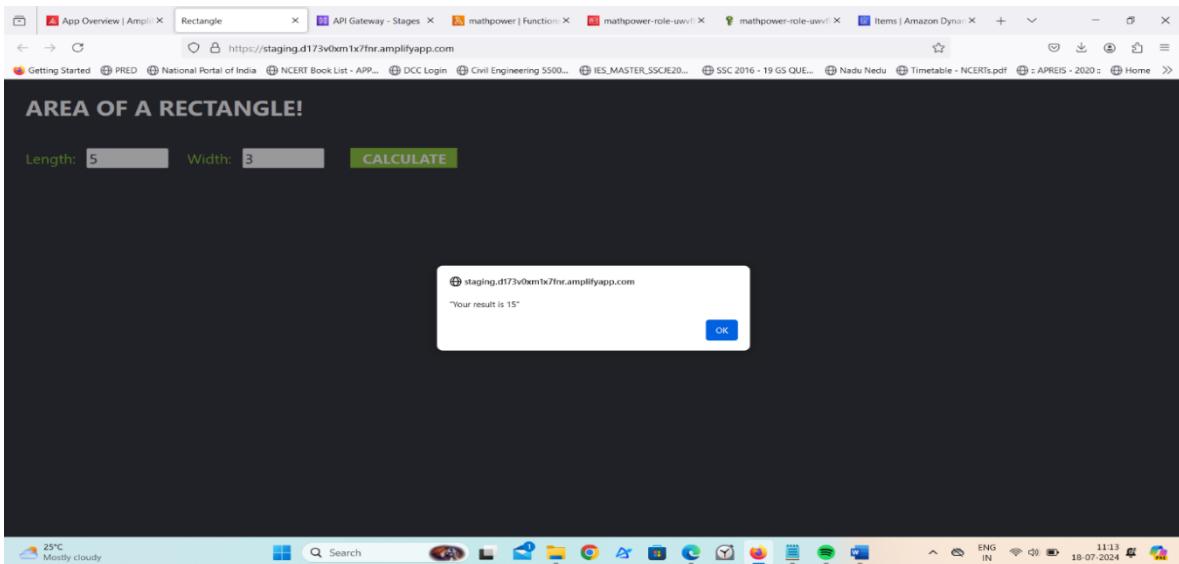


Now go to amplify and click on domain.

Give input values in the length and width space. click on calculate, It will calculate the area of rectangle.

The result is shown at their only.





NIFTY +0.08%

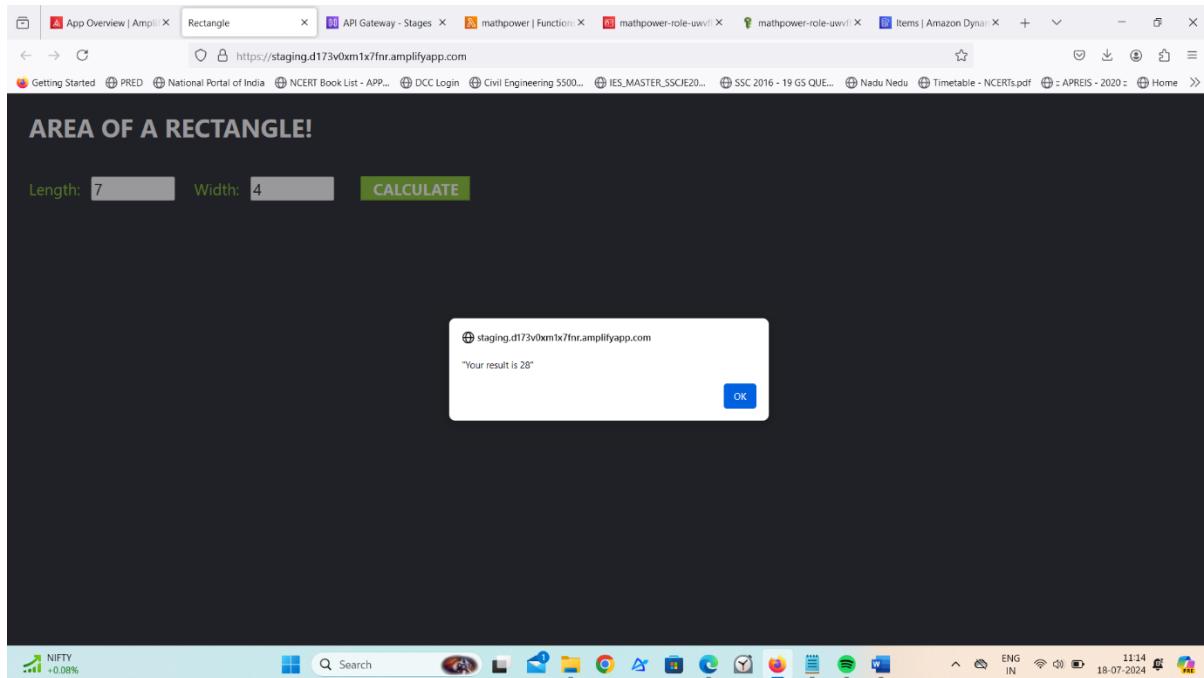
DynamoDB

Completed. Read capacity units consumed: 0.5

ID (String)	LatestGreetingTime
18	Thu, 18 Jul 2024 05:35:59 +0000
8	Thu, 18 Jul 2024 05:35:59 +0000
6	Thu, 18 Jul 2024 04:54:56 +0000
12	Thu, 18 Jul 2024 05:35:59 +0000
15	Thu, 18 Jul 2024 05:35:59 +0000

CloudShell   Feedback   NIFTY +0.08%   © 2024, Amazon Web Services, Inc. or its affiliates.   Privacy   Terms   Cookie preferences

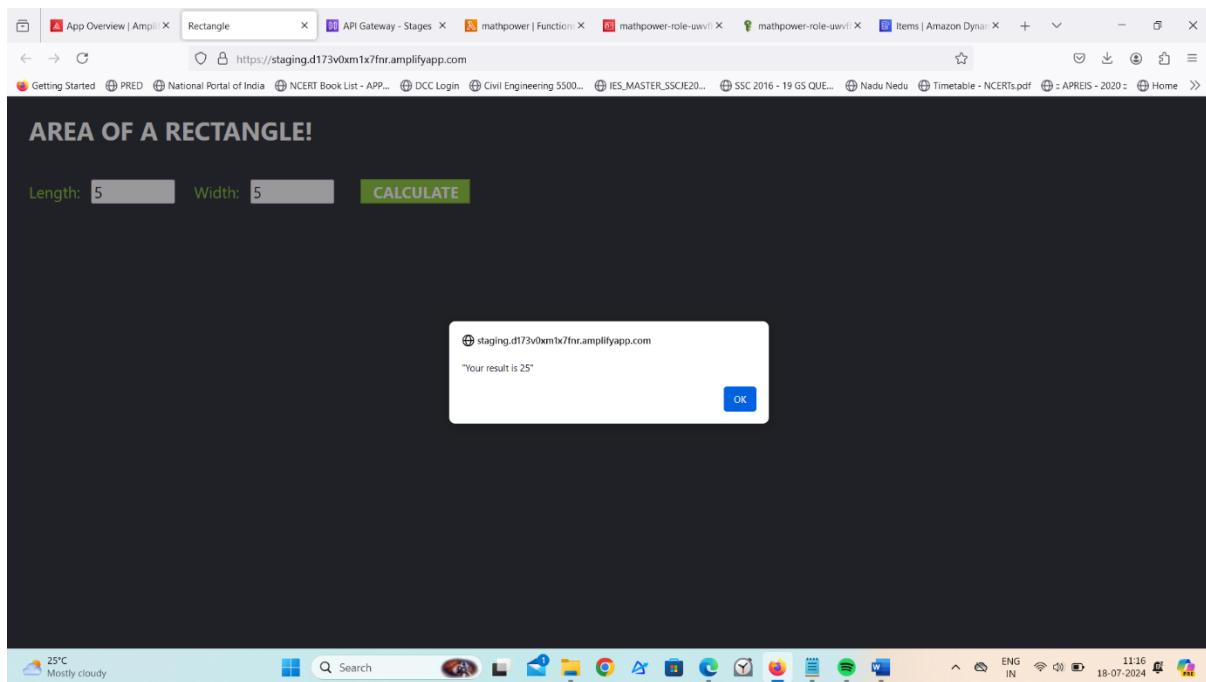
We see that our results are stored in our dynamo DB table.



The screenshot shows the AWS DynamoDB console at the URL <https://us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#item-explorer?fromTables=true&maximize=true&operation=SCAN>. The left sidebar shows the "DynamoDB" navigation pane with options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. The main area shows a "Filters" section with a "Run" button and a "Reset" button. A success message "Completed. Read capacity units consumed: 0.5" is displayed. Below it, a table titled "Items returned (1/6)" lists six items:

ID (String)	LatestGreetingTime
18	Thu, 18 Jul 2024 05:35:59 +0000
8	Thu, 18 Jul 2024 05:35:59 +0000
6	Thu, 18 Jul 2024 04:54:56 +0000
28	Thu, 18 Jul 2024 05:35:59 +0000
12	Thu, 18 Jul 2024 05:35:59 +0000
15	Thu, 18 Jul 2024 05:35:59 +0000

The status bar at the bottom shows "CloudShell Feedback" and the date "18-07-2024".



The screenshot shows the AWS DynamoDB console with the title bar 'aws Services Search [Alt+S]'. The left sidebar is titled 'DynamoDB' and includes options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. The main content area shows a table titled 'Items returned (1/7)'. The table has two columns: 'ID (String)' and 'LatestGreetingTime'. The data rows are:

ID (String)	LatestGreetingTime
18	Thu, 18 Jul 2024 05:35:59 +0000
8	Thu, 18 Jul 2024 05:35:59 +0000
5	Thu, 18 Jul 2024 04:54:56 +0000
28	Thu, 18 Jul 2024 05:35:59 +0000
<b>25</b>	Thu, 18 Jul 2024 05:35:59 +0000
12	Thu, 18 Jul 2024 05:35:59 +0000
15	Thu, 18 Jul 2024 05:35:59 +0000

A success message 'Completed. Read capacity units consumed: 0.5' is displayed above the table. The status bar at the bottom shows the weather as '25°C Mostly cloudy' and the date/time as '18-07-2024 11:17'.

Destroy/delete all created services

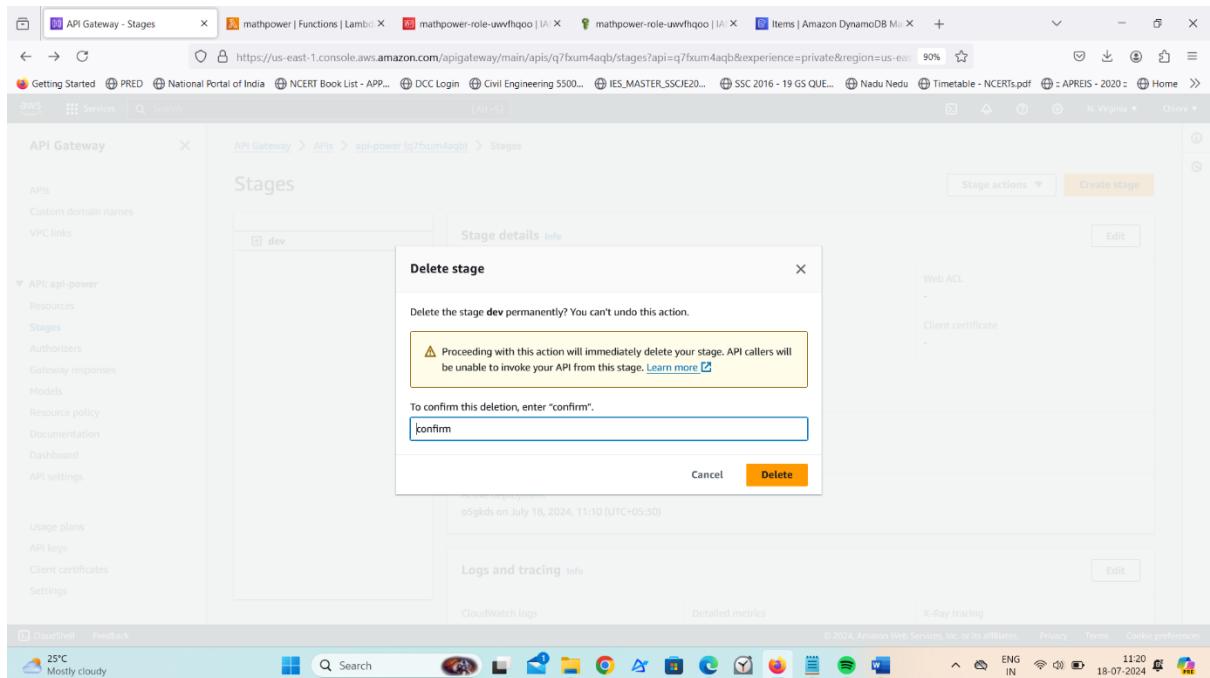
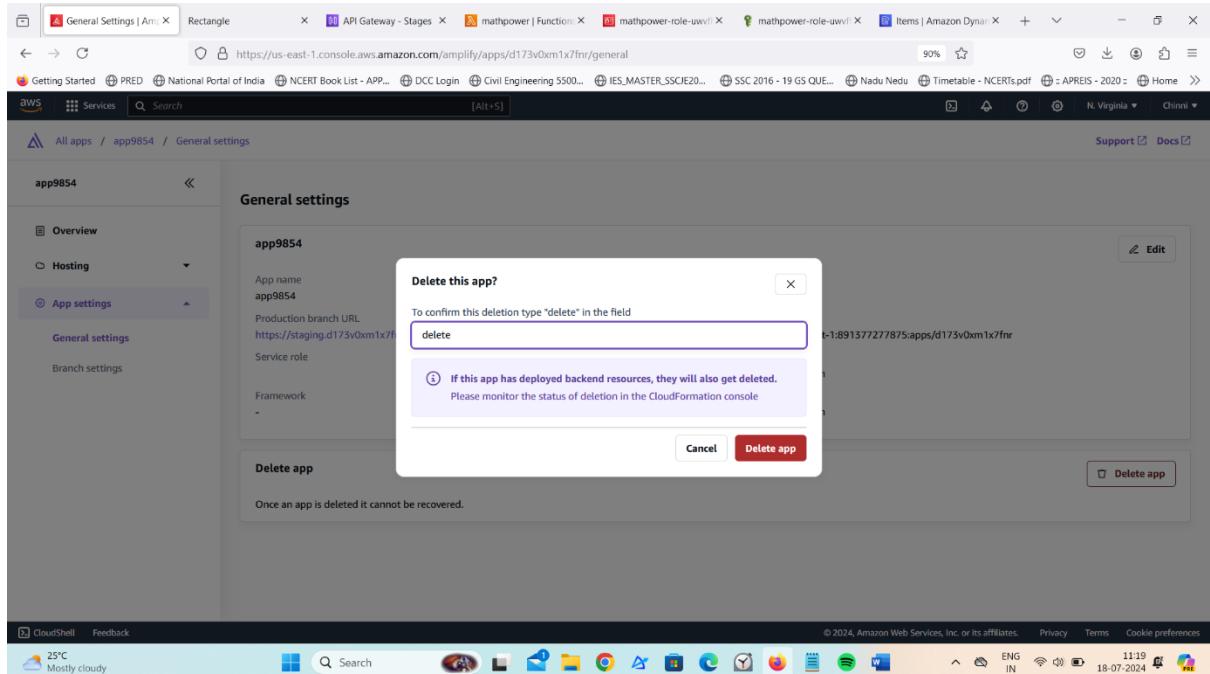
Remember to delete your resources to avoid unnecessary charges:

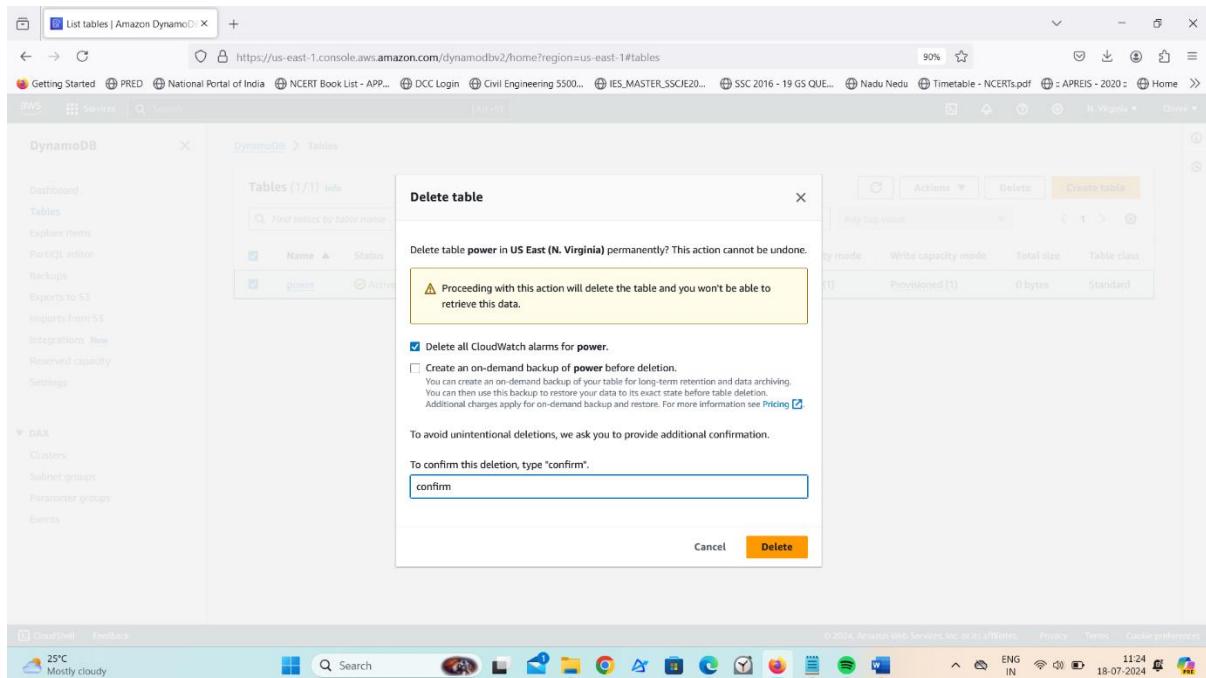
Delete the Amplify App

Delete the DynamoDB Table

## Delete the Lambda function

## Delete the API Gateway





## Conclusion:

In this guide, we've shown how AWS Amplify, DynamoDB, Lambda, and API Gateway can make serverless web app development easy and efficient. By using these tools, you can build powerful applications without worrying about managing servers.

AWS Amplify helps you quickly set up authentication, storage, and APIs. DynamoDB provides a fast and scalable database. Lambda lets you run your code in the cloud without managing servers, and API Gateway makes it simple to create and manage APIs.

Using these services together, you can create web applications that are scalable, cost-effective, and easy to maintain. With the knowledge from this guide, you're ready to start building serverless applications that can grow with your needs. Enjoy the simplicity and power of serverless development with AWS!